Multi-strategy Differential Evolution

Yaman, Anil; Iacca, Giovanni; Coler, Matthew; Fletcher, George; Pechenizkiy, Mykola

Published in:
Applications of Evolutionary Computation

DOI:
10.1007/978-3-319-77538-8_42

Multi-strategy Differential Evolution

Anil Yaman[1(✉)], Giovanni Iacca[2], Matt Coler[3], George Fletcher[1], and Mykola Pechenizkiy[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{a.yaman,g.h.l.fletcher,m.pechenizkiy}@tue.nl
[2] RWTH Aachen University, Aachen, Germany
giovanni.iacca@gmail.com
[3] University of Groningen/Campus Fryslân, Leeuwarden, The Netherlands
m.coler@rug.nl

Abstract. We propose the Multi-strategy Differential Evolution (MsDE) algorithm to construct and maintain a self-adaptive ensemble of search strategies while solving an optimization problem. The ensemble of strategies is represented as agents that interact with the candidate solutions to improve their fitness. In the proposed algorithm, the performance of each agent is measured so that successful strategies are promoted within the ensemble. We propose two performance measures, and show their effectiveness in selecting successful strategies. We then present three population adaptation mechanisms, based on sampling, clone-best and clone-multiple adaptation schemes. The MsDE with different performance measures and population adaptation schemes is tested on the CEC2013 benchmark functions and compared with basic DE and with Self-Adaptive DE (SaDE). Our results show that MsDE is capable of efficiently adapting the strategies and parameters of DE and providing competitive results with respect to the state-of-the-art.

Keywords: Continuous optimization · Differential evolution
Parameter control · Strategy adaptation

1 Introduction

Evolutionary algorithms (EAs) are meta-heuristic search algorithms that operate on a population of candidate solutions. Biologically inspired evolutionary operators -namely selection, mutation and crossover- are used to manipulate iteratively the candidate solutions to improve their fitness [1]. Among EAs, Differential Evolution (DE) has been shown to be an efficient method for several optimization problems [2]. Various kinds of strategies have been suggested in the literature for improving upon basic DE [3,4]: these strategies typically adapt, according to some logics, the mutation scale factor (F) and crossover rate (CR) [5]. Such strategies significantly influence the behavior of DE as they alter the balance between exploration and exploitation [6].

K. Sim and P. Kaufmann (Eds.): EvoApplications 2018, LNCS 10784, pp. 617–633, 2018.
https://doi.org/10.1007/978-3-319-77538-8_42

An appropriate strategy and parameter setting of an algorithm is the best or near-best of all possible settings. Finding an appropriate strategy and parameter setting is an optimization problem that is as hard as finding the solution to the problem [7–9]. Eiben *et al.* categorized the parameters setting problem into two main categories, parameter tuning and parameter control [7]:

1. *Parameter tuning* aims to find the appropriate parameter settings offline, before an evolutionary run. The parameter tuning process can be performed by trial and error, from studies in the literature [10,11], or by using settings of similar problems [12].
2. *Parameter control* on the other hand, aims to adjust the parameter settings during an evolutionary process because the goodness of a parameter setting varies depending on the state of the search [6]. *Deterministic*, *adaptive* and *self-adaptive* methods have been proposed for the parameter control task [7].

We propose here a Multi-strategy Differential Evolution (MsDE) approach to self-adapt strategies and their parameters in DE during an evolutionary process. Most of the self-adaptive parameter control approaches aim to adapt algorithm parameters by including them within the genotype of the individuals and inheriting with the successful individuals during an evolutionary run. In MsDE, distinct from the inheritance based methods, an ensemble of search strategies are employed to operate on, and co-evolve with the candidate solutions. The strategies are referred as agents to distinguish them from the candidate solutions, and underline their function. The agent-based representation of the strategies provides the flexibility to apply a wide range of population adaptation mechanisms. In this work, we present three population adaptation schemes (sampling-based, clone-best and clone-multiple), to show how various self-adaptive agent-based approaches perform on the CEC2013 benchmark functions. Notably, the approach we propose here can be easily extended to any evolutionary algorithm to adapt their operators and parameters.

The rest of the paper is organized as follows: in Sect. 2, we provide the related work, where we discuss the basic DE algorithm and some strategy and parameter adaptation mechanisms proposed in the literature for DE; in Sect. 3, we describe our algorithm, MsDE, and present the three mechanisms for population adaptation; in Sect. 4, we present our test results, with the different population adaptation schemes, on the CEC2013 benchmark functions; finally, in Sect. 5 we provide the conclusions of this work.

2 Related Work

The DE algorithm is a population-based search algorithm proposed for continuous optimization [2]. A candidate solution set $\{x_1, x_2, \ldots, x_{NP}\}$ with a population size of NP is represented as D-dimensional real-valued vectors $x_i \in \mathbb{R}^D, i = 1, 2, \ldots, NP$. In the initialization phase of the algorithm, the candidate solutions are randomly sampled within the domain boundaries of each dimension $j = 1, 2, \ldots, D$.

The algorithm employs a strategy composed of a mutation, crossover, and selection operators with their specified parameters. For each generation g, a candidate solution x_i^g, called *target vector*, is selected $\forall i \in \{1, 2, \ldots, NP\}$. The mutation, crossover and selection operators are then applied to generate a *trial vector* u_i^g, and replace the target vector. The *mutation operator* generates a *mutant vector* v_i^g by perturbing the target vector x_i^g using the scaled differences of several distinct individuals selected randomly from the population. The *crossover operator* generates a trial vector u_i^g by performing recombination between the target vector and the mutant vector. The *selection operator* replaces the target vector x_i^g in the population with the trial vector u_i^g if the fitness value of u_i^g is better than or equal to x_i^g. This process is iteratively executed until a stopping criteria is met.

The **mutation operator** is controlled by the parameter *scale factor* (F) that is used to adjust the magnitude of the perturbation. There are various mutation operators suggested in the literature [3,4]. Four types of mutation strategies, referred as *"DE/rand/1"*, *"DE/rand/2"*, *"DE/rand-to-best/2"*, and *"DE/current-to-rand/1"* are provided in Eqs. (1), (2), (3), and (4), respectively, see [5].

$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{1}$$

$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) + F \cdot (x_{r_4}^g - x_{r_5}^g) \tag{2}$$

$$v_i^g = x_i^g + F \cdot (x_{best}^g - x_i^g) + F \cdot (x_{r_1}^g - x_{r_2}^g) + F \cdot (x_{r_3}^g - x_{r_4}^g) \tag{3}$$

$$v_i^g = x_i^g + K \cdot (x_{r_1}^g - x_i^g) + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{4}$$

where $r_1, r_2, r_3, r_4,$ and r_5 are mutually exclusive integers different from i, and selected randomly from the range $[1, NP]$; the parameter K is a random number uniformly sampled in $(0, 1]$; x_i^g is the target vector; x_{best}^g is the best individual at generation g in terms of fitness.

The **crossover operator** is used to recombine the target vector and the mutant vector with a certain rate, CR, to generate a trial vector u_i^g. The *binomial (uniform) crossover* operator is given in (5). There are several more existing crossover operators such as the *exponential crossover* [13].

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } rand([0,1)) \leq CR \text{ or } j = randi([1, D]); \\ x_{i,j}^g, & \text{otherwise.} \end{cases} \tag{5}$$

where j is an integer within the range $[1, D)$, functions $rand()$ and $randi()$ return a real and an integer value uniformly sampled from a defined range, respectively. The notation $x_{i,j}^g$ refers to the jth dimension of ith vector in the population at generation g.

If the value of the trial vector along the jth dimension exceeds the boundaries defined as x_j^{min} and x_j^{max}, it is randomly and uniformly sampled within the domain boundary range [5], using a toroidal boundary condition [14].

The **selection operator** determines whether or not the trial vector is kept for the next generation $g + 1$. If the fitness value of the trial vector is better

than or equal to the target vector, then the target vector is replaced by the trial vector as it is shown in Eq. (6), which assumes a minimization problem:

$$x_i^{(g+1)} = \begin{cases} u_i^g, \text{ if } f(u_i^g) < f(x_i^g); \\ x_i^g, \qquad \text{otherwise.} \end{cases} \tag{6}$$

The selection phase can be performed synchronously or asynchronously. In synchronous selection, the selected trial vectors are stored in a temporary set, and replaced with target vectors after the selection process of all individuals is complete. In asynchronous selection, a selected trial vector is replaced directly with the target vector without waiting the selection procedure for all individuals. Asynchronous selection makes it possible to use a newly generated trial vector in the trial vector generation process of all the remaining target vectors within the same generation.

2.1 Strategy and Parameter Control in DE

In this section, we highlight the recent developments in strategy and parameter control for DE. Modern variants of DE aim to employ adaptive mechanisms to adjust the algorithm's parameters during an evolutionary run, or across different problems. Strategy and parameter control in DE can be examined in two broad classes: *both strategy and parameter* control, and *only parameter* control [3], where the parameters involved are F and CR. There are also methods for adapting the population size NP, see e.g. [15]; however, in this work we limit our scope to the methods for adapting the strategies, and the parameters F and CR. In the following we briefly describe four of the main DE variants falling in this category, namely EPSDE, SaDE, JADE and jDE.

In the Ensemble of Parameters and mutation Strategies Differential Evolution (EPSDE), mutation strategy and parameter pools are used [16–18]. Each individual in the candidate solution population is assigned with a strategy and a parameter setting from these pools. The strategies and their parameters are inherited from the target to trial vectors as long as they are successful in generating a better trial vector. Otherwise, the strategy and parameters that are associated with the target vector are reinitialized by either randomly sampling from their respective pools, or assigning a strategy and its parameters from the set where successful strategies and parameters are stored.

Self-adaptive differential evolution (SaDE) uses only two mutation strategies, namely *"DE/rand/l/bin"* and *"DE/rand-to-best/1"*, and adapts the parameters F and CR [19]. The strategies and parameters are selected for their properties of generating diverse individuals and faster convergence rate respectively. For each generation, a mutation strategy is randomly selected based on its probability of generating a trial vector successfully. The success probability of the two mutation strategies is initialized uniformly and updated after each generation, based on the number of individuals generated successfully. The scale factor F is randomly sampled, for each individual, from the normal distribution with mean 0.5 and standard deviation 0.3. The parameter CR is initialized for each individual from

a normal distribution with mean 0.5 and standard deviation 0.1. The strategy pool of the SaDE has been later extended by Qin *et al.* [5].

The JADE algorithm introduces a new mutation strategy called "DE/current-to-pbest" with optional archive, and controls the parameters F and CR [20]. The optional archive keeps track of recently explored worse solutions, to provide additional information for the progression of the search. At each generation, the crossover rate CR_i is independently initialized from a normal distribution. The mean of the normal distribution μ_{CR} is initialized as 0.5 in for the first generation, and updated based on the mean of the CR_i of the trial vectors that are generated successfully. The mutation factor F_i is generated and updated in similar fashion by using a Cauchy distribution.

Finally, in jDE [21] the mutation and crossover parameters F and CR are attached to the genotype of the individuals in the population. The algorithm is based on the idea that the parameters that survive with the individuals are likely to produce successful trial vectors; thus, the parameters of the target vectors are propagated to the successive trial vectors in the next generations.

3 Multi-strategy Differential Evolution (MsDE)

The MsDE aims to self-adapt the strategy types (mutation and crossover operators) and their parameters (F and CR) used in DE while solving the optimization problem. It employs an ensemble of strategies with certain parameter settings, and applies population adaptation schemes to construct and maintain the ensemble strategy set. Different from the established ensemble methods in the literature, the MsDE considers the strategies as agents that interact with the candidate solution set. The agent-based representation of the strategies provides the basis for an easy application of population adaptation approaches.

The pseudocode of MsDE (assuming an asynchronous population, see below) is provided in Algorithm 1. The algorithm takes NP (number of solutions) and m (number of strategies) as parameters. In addition, there are two thresholds we refer to as performance and maturation thresholds, τ and δ, for determining the performance of a strategy and limiting the test phase of a strategy. The performance threshold τ is an adaptive threshold based on the average value of all the performances in the strategy ensemble. The maturation threshold δ is typically a small integer (e.g. 5) used to control how many algorithm iterations should be invested for the testing phase of new strategies.

The candidate solution set X consisting of NP D-dimensional real-valued vectors $x_i \in \mathbb{R}^D$ that represent a solution to the problem. The initial candidate solutions are randomly sampled in the domain range for each dimension. The population size NP is chosen during the initialization phase, and remains fixed throughout the run.

The ensemble strategy set Σ consists of m strategies $\sigma_1, \sigma_2, \ldots, \sigma_m \in \Sigma$. Each σ_j defines a kind of mutation and crossover operator, with specified parameters F and CR. In the initialization phase, each strategy is initialized by selecting a random mutation strategy with a type of crossover operator from a predefined

Algorithm 1. Asynchronous MsDE

procedure MsDE(NP, m)
  $g \leftarrow 0$ ▷ generation count
  initialize X ▷ randomly initialize NP solutions
  $\forall \sigma_j \in \Sigma, j = 1, 2, \ldots, m; \sigma_j \leftarrow InitializeRandomStrategy()$ ▷ see Alg. 2
  $F \leftarrow evaluate(X)$
  while termination criterion is not satisfied **do**
    $\tau \leftarrow mean(P_\Sigma)$ ▷ τ is the average performance of the strategies
    for each $\sigma \in \Sigma$ **do**
      $targetVector \leftarrow randSelect(X)$ ▷ randomly select a target vector
      $mutantVector \leftarrow \sigma.mutate(targetVector)$
      $trialVector \leftarrow \sigma.crossover(targetVector, mutantVector)$
      $\sigma.totalActivation \leftarrow \sigma.totalActivation + 1$
      $F_{trial} \leftarrow evaluate(trialVector)$
      if $F_{trial} < F_{target}$ **then** ▷ selection operator (assuming minimization)
        $targetVector \leftarrow trialVector$
        $F_{target} \leftarrow F_{trial}$
        $\sigma.successfulActivation \leftarrow \sigma.successfulActivation + 1$
      end if
      $P_{\sigma_j} \leftarrow evaluate(\sigma)$ ▷ performance of a strategy
      if $P_{\sigma_j} < \tau$ and $\sigma.totalActivation > \delta$ **then**
        reinitialize σ
      end if
    end for
    $g \leftarrow g + 1$
  end while
end procedure

set of strategies S, of size l. The parameters F and CR are randomly sampled from a uniform distribution in $(0, 1.2]$ and $[0, 1]$, respectively. The upper limit of the scale factor is set to 1.2 because of the works that report the effective range for F between $(0, 1.2]$ [16]. The initialization procedure is illustrated in Algorithm 2.

Algorithm 2. Initialize random strategy

function InitializeRandomStrategy()
  $randomIndex \leftarrow randi[1, l]$
  $\sigma_{random}.type \leftarrow S[randomIndex]$
  $\sigma_{random}.F \leftarrow rand(0, 1.2]$
  $\sigma_{random}.CR \leftarrow rand[0, 1]$
  return σ_{random}
end function

The main loop of MsDE repeats until a stopping criteria is reached. In each iteration, each strategy agent σ_j is executed $\forall j \in (1, 2, \ldots, m)$, such that: first,

a target vector x_i from X is randomly selected; secondly, the mutation and crossover operators are applied to generate a trial vector u_i; finally, the selection operator is applied to replace the target vector with the trial vector if its fitness value is better or equal. The selection operator can be *synchronous* or *asynchronous* as discussed in Sect. 2.

The MsDE calculates a performance measure within the function $evaluate(\sigma)$ to evaluate each strategy. Based on this performance measure, the strategies are classified as successful or unsuccessful. To solve the problem efficiently, firstly successful or unsuccessful strategies should be identified as quickly as possible; and secondly, the number of successful strategies in the population should be maximized, or, vice versa, the number of unsuccessful strategies should be minimized. We discuss these two aspects in the following sections.

Identifying Successful Strategies. Constructing and maintaining a successful set of strategies is crucial for the performance of the algorithm. The self-adaptive mechanism for ensemble construction and maintenance should be capable of managing the trade-off of exploring and exploiting successful strategies efficiently and adaptively. We use a performance measure to asses the quality of a strategy. We propose two measures P_1 and P_2 with different properties. The performance measure P_1, given in Eq. (7), measures the ratio between the number of strategy activations that led to a successful action and the total number of strategy activations:

$$P_1 = \frac{\sigma_j.successfulActivation}{\sigma_j.totalActivation} \tag{7}$$

where $\sigma_j.successfulActivation$ is the number of activations in which a strategy σ_j produced a trial vector with better fitness than the target vector, and $\sigma_j.totalActivation$ is the number of total activations.

As we will demonstrate in Sect. 4, a strategy selection criterion based on P_1 is likely to facilitate fast convergence; the drawback is a high probability of getting stuck onto a local optimum if the function is multimodal. This is because the strategies that are exploitative are more likely to score higher on P_1 than the exploratory strategies because small exploitative increments on the solutions are more likely to yield better solutions. To prevent the domination of exploitative strategies in the ensemble, the performance measure should be improved to promote also exploratory strategies. We measure the exploratory value of a strategy by its capability to produce diverse individuals with better fitness values. Such performance evaluation criteria would also encourage the diversity in the population; thus, it may be less susceptible to early convergence.

Methods used in multi-objective optimization, such as non-dominated sorting, can be used to select the strategies that have diverse trial vectors generation rate and high ratio of success [22]. On the other hand, these methods can increase the complexity of the algorithm. Thus, to avoid further complexity, we provide a performance measure P_2 for a single selection criterion to combine these two aspects implicitly in Eq. (8), where we calculate the average differences between

target and trial vectors for the last γ activations in which the trial generation was successful.

$$P_2 = \begin{cases} \frac{1}{\sum_{a=TA_{\sigma_j}-\gamma+1}^{TA_{\sigma_j}} \psi_{\sigma_j}^{(a)}} \cdot \sum_{a=TA_{\sigma_j}-\gamma+1}^{TA_{\sigma_j}} \Delta_{\sigma_j}^{a} \cdot \psi_{\sigma_j}^{(a)}, \text{ if } TA_{\sigma_j} \geq \gamma; \\ \frac{1}{\sum_{a=1}^{TA_{\sigma_j}} \psi_{\sigma_j}^{(a)}} \cdot \sum_{a=1}^{TA_{\sigma_j}} \Delta_{\sigma_j}^{a} \cdot \psi_{\sigma_j}^{(a)}, \qquad \text{otherwise.} \end{cases} \tag{8}$$

$$\Delta_{\sigma_j}^{(a)} = \sum_{d=1}^{D} \mid x_{i,d}^{(a)} - u_{i,d}^{(a)} \mid \tag{9}$$

$$\psi_{\sigma_j}^{(a)} = \begin{cases} 1, \text{ if } f(x_i^{(a)}) < f(u_i^{(a)}); \\ 0, \qquad \text{otherwise.} \end{cases} \tag{10}$$

where $\Delta_{\sigma_j}^{(a)}$, defined by the distance metric given in Eq. (9), is the sum of the absolute differences along each dimension between target and trial vectors, and TA_{σ_j} represents $\sigma_j.totalActivation$. The parameter γ is introduced into this measure to sum only the differences in the most recent history of activations, to have a self-adaptive property. If γ is a large number then the measure may promote exploratory strategies that may have a few successful activations with large diversity.

Maximizing the Number of Successful Strategies. For an efficient search, the strategies that are classified as successful should be kept in the ensemble as long as they remain successful. To identify the performance of a strategy, strategies are tested for a certain time. The testing phase consumes resources, namely function evaluations (FEs). Since the strategies activated during the testing phase may not be necessarily good, the resources consumed in this phase should be minimized.

To distinguish the successful and unsuccessful strategies, the average performance values of all strategies τ is used. If the performance value of a strategy $P_{\sigma_j} < \tau$, then it is considered to be unsuccessful. To collect necessary evidence on the performance of a strategy, a *maturation threshold* δ is used such that if a strategy does not exceed δ then it is neither classified as successful nor unsuccessful.

3.1 Strategy Population Adaptation Schemes

We propose three mechanisms for strategy population adaptation: sampling-based, clone-best, and clone-multiple population adaptation schemes. These population adaptation schemes provide the logic for initiating new strategies into the ensemble, and removing existing strategies from the ensemble. The population adaptation methods are implemented in Line 21 of Algorithm 1.

Sampling-Based Population Adaptation. The sampling-based adaptation scheme initiates new strategies based on random sampling. The sampling function given in Algorithm 2 is used to reinitialize a mature unsuccessful strategy.

Clone-Best Population Adaptation. The clone-best adaptation scheme implements a clonal reproduction mechanism to replace unsuccessful strategies. The clone-best scheme is inspired by the clonal selection principle of the immune system theory [23,24]. The main idea is to replace an unsuccessful strategy with a clone of the best performing strategy with a small perturbation.

Algorithm 3. Strategy reinitialization by clone-best

```
1: function Clone(σ_j^g)
2:     if rand(0,1) < φ then
3:         if rand(0,1) < η then
4:             σ_clone^g.type ← S[randomIndex]
5:         else
6:             σ_clone^g.type ← σ_j^g.type
7:         end if
8:         σ_clone^g.F ← σ_j^g.F + η · N(0,1)
9:         σ_clone^g.CR ← σ_j^g.CR + η · N(0,1)
10:    else
11:        σ_clone ← InitializeRandomStrategy()
12:    end if
13:    return σ_clone
14: end function
```

Algorithm 3 shows the function that is used to clone a strategy. In clone-best, the function takes σ^g_{best} as an argument, where σ^g_{best} is the best strategy in the current generation g. With probability ϕ, the *type*, (F and CR) of an unsuccessful strategy is replaced by the *type* (F, and CR) of the best strategy in the ensemble, with a small perturbation with scale factor $\eta \in (0, 1]$. In our experiments, we use $\eta = 0.1$. If the parameter boundaries are exceeded, they are reinitialized by a value close to the boundaries. If $rand(0, 1) \geq \phi$ the strategy is reinitialized using the uniform sampling scheme given in Algorithm 2.

Clone-Multiple Population Adaptation. The clone-multiple adaptation is an extension of the clone-best adaptation where m successful strategies are kept in a separate set referred to as memory strategies (Σ). If the limit of Σ is not exceeded, the scheme aims to find more successful strategies to add to Σ by going through the *cloning*, *selection*, *maturation* and *promotion* phases. These phases are described below:

1. ***Clonal expansion:*** n best strategies from Σ are selected and assigned into the best strategy set B. Each strategy in B is cloned (with a small perturbation) proportional to their performance. The higher the performance of a

strategy, the higher the number of clones generated. Algorithm 3, without ϕ parameter (or $\phi = 1$), is used for generating each clone for each $\sigma_j^g \in B$. Generated clones are added to a temporary candidate clone set T.

2. ***Clonal selection:*** h ($h \leq n$) candidate clones from T are selected based on their similarity to the strategies in B; and $v (v \leq h)$ strategies are generated randomly. The selected and randomly generated individuals are then added to the clone set C, that has size $h + v$. The similarity-based clone selection and the random strategy generation criteria are executed as follows:
   - h individuals are selected as follows: for each $\sigma_i \in T, i = 1, 2, \ldots, size(T)$ and $sigma_j \in B, j = 1, 2, \ldots, n$, $d_{\sigma_i} = \sum_{j=1}^{n} dist(\sigma_i, \sigma_j)$ is calculated. The h candidate clones with smallest d_{σ_i} are added to the clone set C. The $dist(\sigma_i, \sigma_j)$ computes the Euclidean distance between the parameters (F and CR) of σ_i and σ_j;
   - v random strategies are generated using Algorithm 2.
3. ***Maturation:*** each strategy in C is tested for δ FEs.
4. ***Promotion:*** strategies in C that are successful (i.e., that satisfy the performance threshold) are added to the memory set Σ. The same classification criterion for finding unsuccessful/successful strategies is used for finding successful strategies in C.

The logic behind the clone-multiple population adaptation scheme is such that it reduces the trial and error of newly generated strategies, by cloning successful strategies that are kept in a separate set. It also aims to find strategies that are likely to perform well by making a similarity-based selection. Mutations during the cloning phase allow for exploration of different strategies with different parameter settings.

4 Experimental Setup and Results

In this section, we present our experimental results on the CEC2013 benchmark functions [25]. The objective of our experiments is threefold. First, we show the effect of the two strategy performance measures P_1 and P_2 proposed in Sect. 3 on MsDE, with three population adaptation schemes. Second, we illustrate how the strategy adaptation dynamics compare between the sampling, clone-best and clone-multiple based population adaptation schemes during an evolutionary run. Finally, we compare the MsDE with basic DE and SaDE.

The types of the strategies and parameters of the algorithms are the same for all the experiments, unless otherwise specified. We employ four types of DE strategies referred to as *"DE/rand/1/bin"*, *"DE/rand/2/bin"*, *"DE/rand-to-best/2/bin"*, and *"DE/current-to-rand/1"*. The suffix *"bin"* refers to the binomial crossover. Note that *"DE/current-to-rand/1"* does not include a crossover operator. These strategies are selected on the basis of previous comparisons performed in the literature; furthermore, they are also used in SaDE [5]. Asynchronous selection is used for the selection operator.

All the experiments were performed using $NP = 100$ candidate solutions and $m = 50$ strategies. The algorithms were run for at most $5000 \times D$ function

evaluations (FEs), where D is the dimension of the problem. If the error between the best solution found and the global optimum is less than or equal to $1e-8$, we terminate the algorithm. Each algorithm was executed for 25 independent runs; the mean and standard deviation of minimum error $f(x_{best}) - f(x^*)$ achieved are presented.

The three strategy adaptation schemes (sampling-based, clone-best and clone-multiple) are referred to as *MsDE-Sam*, *MsDE-CB*, and *MsDE-CM*, respectively. For MsDE-CB, the probability for cloning the best strategy ϕ is set to 0.7. For MsDE-CM, the number of selected best strategies n is set to 10, the max number of clones per strategy is set to 10 for the best strategy and reduced by 1 per each lower ranked strategy, the number of similar selected strategies is set to $h = 7$, and the number of randomly initialized strategies is set to $v = 3$; thus the number of strategies adds up to a total number of 10. For all algorithms, the maturation threshold and the history threshold γ are set to 5 FEs and 10 activations, respectively.

Comparing the Performance Measures. In Table 1, we compare the two different performance measures P_1 and P_2 for each kind of population adaptation scheme on the CEC2013 functions in 10 dimensions. The suffixes "$-P_1$" and "$-P_2$" indicate the performance measure used with a specific kind of population adaptation scheme. The best results for each performance measure for each algorithm setting is highlighted in bold. The results that do not have significant difference were not highlighted. The global best result for each function is marked by the symbol "*****".

We observed that all three population adaptation schemes perform significantly better using P_2 on almost all benchmark functions. Since P_1 promotes the strategies based solely on the ratio of producing successful trial vectors, it is likely to promote exploitative strategies that can cause early convergence, or stalling the progress with small improvements. Performance measure P_2, on the other hand, promotes strategies that can produce diverse trial vectors successfully. The rest of the experiments are performed using P_2.

Strategy Ensemble Adaptation Dynamics. Next, we examine how the strategies adapt over time. In Fig. 1, we provide the results on f_2 (first column) and f_6 (second column) in 30 dimensions.

Each sub-figure in Fig. 1 shows how the distribution of strategies changes during an evolutionary run. Each line in the figures represents the number of strategies of a given type in the strategy population, at a given generation. Only the strategies that are mature and above the success threshold are counted. We observe that in MsDE-Sam (a) and (d), there is a baseline pool of random strategies that explores new strategies. The ratio of these pool is about %30 of the whole population. In MsDE-CB (b) and (e), this ratio is about %20; and in MsDE-CM (c) and (f), we observe that the random strategy pool is almost nonexistent, and there is usually one type of strategy that is dominant at each time. MsDE-Sam scheme constantly explores different strategies by keeping a

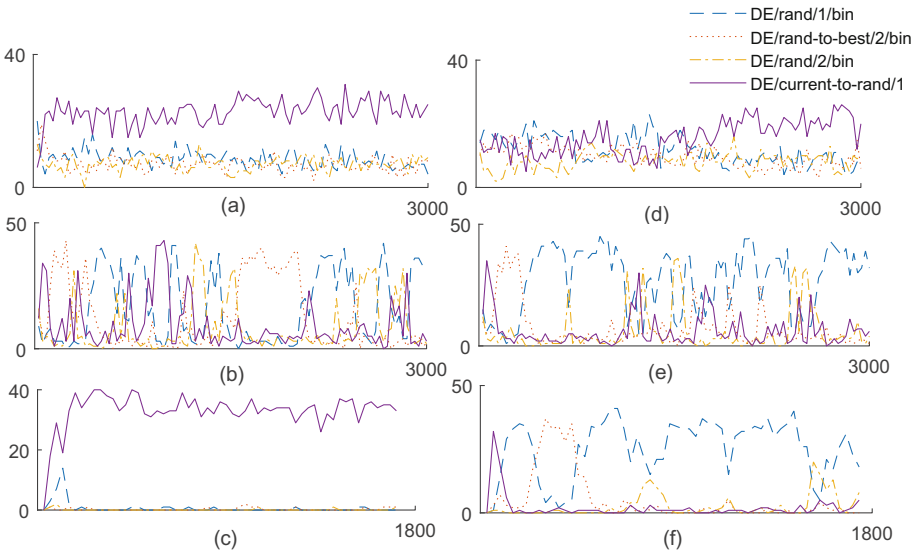

Fig. 1. The distribution of strategies in the strategy population during an evolutionary run. The first and second columns show the results for f_2 and f_6 in 30 dimensions, while the rows show the results of MsDE-Sam, MsDE-CB, and MsDE-CM, respectively.

small set of random strategies which can consume resources (number of function evaluations). On the other hand, MsDE-CB and MsDE-CM aims to exploit already found successful strategies by reintroducing them into the ensemble by reproduction.

Comparing with Other Algorithms. Finally, we test MsDE-Sam, MsDE-CB, MsDE-CM, basic DE and SaDE [5] on the CEC2013 benchmark functions in 30 dimensions. The parameters F and CR of the basic DE set are as 0.5 and 0.3 respectively. The results are given in Table 2. The global best result for each function is highlighted in bold.

To assess the statistical significance of the results, we perform the Wilcoxon rank-sum test [26] based on the results provided in Table 2. The Wilcoxon rank-sum test is a non-parametric test that does not assume normality condition [14,26]. It is a pairwise test that aims to detect the significant difference between two different means that are the results of two algorithms. We reject the null-hypothesis, that is the behavior of the two algorithms are the same, if the p-value is smaller than $\alpha = 0.05$. We compare the MsDE-CM with the other algorithms using the best function error values of 25 independent runs for each benchmark function. The results are given in Table 2 next to the columns of the specified algorithms (except MsDE-CM, which is taken as the reference algorithm), where "=", "+" and "−" indicate no significant difference, significant difference in favor of MsDE-CM, and significant difference in favor of the specified algorithm.

Table 1. The experiment results with performance measures P_1 and P_2 on the CEC2013 benchmark problems in 10 dimensions.

f_i	MsDE-Sam-P_1	MsDE-Sam-P_2	MsDE-CB-P_1	MsDE-CB-P_2	MsDE-CM-P_1	MsDE-CM-P_2
f_1	8.47E-09 ± 1.92E-09	8.74E-09 ± 1.15E-09	9.04E-09 ± 1.31E-04	8.52E-09 ± 1.63E-09	8.49E-09 ± 2.31E-02	8.77E-09 ± 5.37E-09
f_2	1.27E+03 ± 4.20E+03	**8.37E+02 ± 3.20E+03**	1.79E+05 ± 3.34E+05	**9.42E-09 ± 3.52E-05***	1.00E+05 ± 3.50E+06	**8.16E-09 ± 1.15E-06***
f_3	5.77E+05 ± 1.03E+07	**2.08E+01 ± 3.99E+05**	1.72E+07 ± 4.99E+08	**4.95E-03 ± 1.28E+00***	3.44E+07 ± 1.06E+09	**1.62E-01 ± 1.25E+00**
f_4	8.19E-03 ± 9.28E-01	**2.51E-03 ± 5.32E-02**	5.45E+02 ± 4.52E+03	**8.83E-09 ± 1.31E-09***	5.36E+01 ± 6.69E+03	**8.66E-09 ± 9.76E-09***
f_5	9.27E-09 ± 1.07E-09	8.65E-09 ± 1.11E-09	9.70E-09 ± 7.56E-02	9.50E-09 ± 6.92E-08	9.02E-09 ± 8.79E-02	8.97E-09 ± 1.16E-07
f_6	**9.53E-09 ± 4.97E+00***	9.81E+00 ± 4.81E+00	1.01E+01 ± 8.31E+00	**8.06E+00 ± 3.67E+00**	9.82E+00 ± 2.66E+01	**8.67E-09 ± 2.72E+00***
f_7	1.63E+01 ± 1.61E+01	**1.53E+00 ± 8.36E+00**	6.13E+01 ± 2.80E+01	**2.04E+01 ± 1.21E+01**	3.13E+01 ± 4.49E+01	**1.21E-03 ± 5.52E-02***
f_8	2.04E+01 ± 7.74E-02	2.04E+01 ± 9.58E-02	2.04E+01 ± 8.67E-02	**3.44E+00 ± 7.08E-02**	2.04E+01 ± 1.22E-01	**2.04E+01 ± 9.15E-02***
f_9	3.78E+00 ± 1.27E+00	**2.92E+00 ± 9.42E-01**	6.60E+00 ± 1.39E+00	**4.43E-02 ± 1.28E+00***	7.41E+00 ± 1.70E+00	**1.96E+00 ± 1.20E+00**
f_{10}	2.78E-01 ± 2.71E-01	**1.40E-01 ± 6.72E-02**	**8.89E-01 ± 2.73E+00**	1.99E+00 ± 4.68E-02	8.82E-01 ± 1.62E+01	**5.88E-08 ± 2.99E-02***
f_{11}	2.98E+00 ± 3.52E+00	**9.34E-09 ± 6.47E-01***	1.39E+01 ± 1.14E+01	**1.09E+01 ± 2.36E+00**	6.59E+00 ± 1.41E+01	**9.15E-09 ± 4.97E-01***
f_{12}	1.59E+01 ± 8.37E+00	**1.09E+01 ± 5.33E+00**	3.28E+01 ± 1.79E+01	**2.44E+01 ± 5.04E+00**	1.88E+01 ± 1.73E+01	**3.98E+00 ± 2.58E+00***
f_{13}	2.65E+01 ± 9.98E+00	**1.58E+01 ± 7.46E+00**	4.66E+01 ± 2.75E+01	**4.33E+01 ± 1.09E+01**	3.57E+01 ± 2.52E+01	**5.05E+00 ± 5.09E+00***
f_{14}	1.51E+01 ± 3.80E+01	**3.54E+00 ± 6.96E+00**	**6.17E+01 ± 1.91E+02**	7.68E+02 ± 1.08E+02	6.48E+01 ± 2.53E+02	**3.12E-01 ± 3.86E+00***
f_{15}	7.96E+02 ± 2.53E+02	**7.02E+02 ± 2.74E+02**	8.19E+02 ± 3.41E+02	**4.24E-02 ± 2.11E+02***	9.42E+02 ± 4.02E+02	**7.83E+02 ± 2.65E+02**
f_{16}	**9.49E-01 ± 3.27E-01**	1.07E+00 ± 3.00E-01	**4.91E-01 ± 2.11E-01***	1.22E+01 ± 5.89E-01	**1.07E+00 ± 4.45E-01**	1.08E+00 ± 6.94E-01
f_{17}	1.04E+01 ± 3.64E+00	**1.02E+01 ± 9.64E-02***	3.15E+01 ± 1.31E+01	**1.98E+01 ± 1.21E+00**	1.44E+01 ± 6.61E+00	**1.03E+01 ± 6.07E-01**
f_{18}	**2.09E+01 ± 4.75E+00**	2.14E+01 ± 5.71E+00	4.02E+01 ± 1.97E+01	**7.42E-01 ± 4.09E+00***	3.04E+01 ± 1.32E+01	**1.60E+01 ± 2.63E+00**
f_{19}	6.00E-01 ± 2.52E-01	**3.68E-01 ± 1.39E-01***	**1.45E+00 ± 1.28E+00**	3.30E+00 ± 2.30E-01	1.02E+00 ± 6.46E-01	**5.13E-01 ± 1.70E-01**
f_{20}	**2.43E+00 ± 5.81E-01**	2.78E+00 ± 5.05E-01	**3.89E+00 ± 5.17E-01**	4.00E+02 ± 6.05E-01	3.55E+00 ± 6.30E-01	**2.40E+00 ± 5.13E-01***
f_{21}	4.00E+02 ± 6.28E+01	4.00E+02 ± 5.54E+01	4.00E+02 ± 2.31E-01	**8.44E+01 ± 2.32E-13***	4.00E+02 ± 8.73E+01	4.00E+02 ± 7.08E+01
f_{22}	7.54E+01 ± 8.53E+01	**3.12E+01 ± 7.26E+01***	**1.68E+02 ± 2.35E+02**	9.33E+02 ± 1.69E+02	3.20E+02 ± 2.79E+02	**4.99E+01 ± 5.09E+01**
f_{23}	9.89E+02 ± 3.11E+02	**8.33E+02 ± 2.40E+02**	7.92E+02 ± 3.28E+02	**2.06E+02 ± 2.66E+02***	1.21E+03 ± 3.45E+02	**8.67E+02 ± 3.07E+02**
f_{24}	2.12E+02 ± 2.64E+01	**2.09E+02 ± 2.56E+01**	2.21E+02 ± 2.49E+01	**2.00E+02 ± 2.06E+01**	2.18E+02 ± 3.52E+01	**2.00E+02 ± 1.62E+01***
f_{25}	2.11E+02 ± 2.18E+01	**2.02E+02 ± 2.28E+01**	2.20E+02 ± 2.17E+01	**1.19E+02 ± 1.74E+01***	2.18E+02 ± 1.98E+01	**2.00E+02 ± 1.78E+01**
f_{26}	**1.56E+02 ± 4.10E+01**	2.00E+02 ± 3.88E+01	**1.32E+02 ± 3.65E+01**	3.00E+02 ± 3.21E+01	1.51E+02 ± 3.54E+01	**1.06E+02 ± 2.67E+01***
f_{27}	4.00E+02 ± 8.74E+01	**3.00E+02 ± 7.35E+01**	4.58E+02 ± 9.41E+01	**3.00E+02 ± 2.76E+01***	4.41E+02 ± 1.09E+02	**3.00E+02 ± 2.00E+01**
f_{28}	3.00E+02 ± 1.43E+02	**3.00E+02 ± 6.63E+01***	**3.00E+02 ± 2.15E+02**	3.00E+02 ± 9.80E+01	**3.00E+02 ± 1.86E+02**	3.00E+02 ± 4.00E+01

Table 2. The experiment results of the selected algorithms on the CEC2013 benchmark problems in 30 dimensions.

f_i	MsDE-Sam		MsDE-CB		MsDE-CM	DE		SaDE	
f_1	9.56E-09 ± 7.88E-10	=	9.26E-09 ± 9.58E-10	=	9.54E-09 ± 6.05E-10	9.24E-09 ± 5.51E-10	=	9.14E-09 ± 1.14E-09	=
f_2	1.40E+05 ± 1.32E+05	=	1.71E+05 ± 1.15E+05	=	**1.19E+05 ± 1.91E+05**	2.21E+08 ± 4.53E+07	+	1.76E+05 ± 1.45E+05	=
f_3	1.15E+05 ± 4.06E+06	=	6.01E+06 ± 1.37E+07	+	**8.60E+04 ± 1.59E+06**	1.56E+09 ± 8.56E+08	+	1.43E+05 ± 1.72E+06	=
f_4	**1.22E+01 ± 3.63E+01**	−	1.73E+02 ± 3.74E+02	+	4.40E+01 ± 6.08E+01	1.16E+05 ± 1.64E+04	+	3.52E+03 ± 2.42E+03	+
f_5	9.58E-09 ± 4.97E-10	=	9.64E-09 ± 7.46E-10	=	9.71E-09 ± 3.71E-10	9.58E-09 ± 5.56E-10	=	9.60E-09 ± 6.51E-10	=
f_6	9.67E+00 ± 5.07E+00	=	9.41E+00 ± 6.67E+00	=	**9.17E+00 ± 1.72E+01**	2.63E+01 ± 3.33E-01	+	1.22E+01 ± 1.25E+01	+
f_7	1.46E+01 ± 1.03E+01	−	7.23E+01 ± 1.64E+01	+	2.87E+01 ± 1.40E+01	1.46E+02 ± 1.48E+01	+	**1.22E+01 ± 6.62E+00**	−
f_8	**2.10E+01 ± 6.72E-02**	−	2.10E+01 ± 4.33E-02	−	2.10E+01 ± 6.37E-02	2.10E+01 ± 5.53E-02	−	2.10E+01 ± 5.50E-02	−
f_9	**1.73E+01 ± 3.08E+00**	−	2.72E+01 ± 4.73E+00	+	2.24E+01 ± 5.74E+00	3.94E+01 ± 1.14E+00	+	1.91E+01 ± 2.32E+00	−
f_{10}	3.45E-02 ± 2.64E-02	=	5.17E-02 ± 3.94E-02	=	**3.45E-02 ± 2.56E-02**	1.27E+02 ± 3.62E+01	+	4.93E-02 ± 4.01E-02	=
f_{11}	8.95E+00 ± 4.54E+00	−	2.98E+01 ± 1.16E+01	+	1.69E+01 ± 8.36E+00	6.83E+01 ± 5.55E+00	+	**1.81E-05 ± 5.57E+00**	−
f_{12}	4.58E+01 ± 9.63E+00	+	7.16E+01 ± 2.25E+01	+	**3.18E+01 ± 8.87E+00**	2.03E+02 ± 1.06E+01	+	3.48E+01 ± 6.34E+00	+
f_{13}	8.77E+01 ± 2.31E+01	+	1.54E+02 ± 4.15E+01	+	**6.86E+01 ± 2.37E+01**	2.10E+02 ± 1.50E+01	+	6.94E+01 ± 2.64E+01	=
f_{14}	**2.90E+01 ± 3.26E+01**	−	9.99E+02 ± 4.87E+02	+	6.11E+01 ± 1.25E+02	3.93E+03 ± 2.33E+02	+	1.81E+03 ± 6.37E+02	+
f_{15}	6.41E+03 ± 1.15E+03	+	3.63E+03 ± 5.78E+02	=	3.68E+03 ± 1.49E+03	7.78E+03 ± 3.10E+02	+	**3.42E+03 ± 5.60E+02**	=
f_{16}	2.59E+00 ± 2.30E-01	+	**2.03E-01 ± 8.17E-01**	−	2.33E+00 ± 9.58E-01	2.71E+00 ± 3.02E-01	+	2.51E+00 ± 3.66E-01	=
f_{17}	**3.34E+01 ± 1.61E+00**	−	6.01E+01 ± 1.11E+01	+	3.57E+01 ± 4.18E+00	1.00E+02 ± 6.01E+00	+	5.80E+01 ± 9.32E+00	+
f_{18}	1.70E+02 ± 4.10E+01	+	8.28E+01 ± 1.76E+01	+	**5.15E+01 ± 3.82E+01**	2.32E+02 ± 1.05E+01	+	5.55E+01 ± 1.15E+01	=
f_{19}	2.70E+00 ± 1.04E+00	+	4.95E+00 ± 2.64E+00	+	**2.40E+00 ± 5.24E-01**	1.10E+01 ± 6.77E-01	+	2.75E+00 ± 9.47E-01	+
f_{20}	1.14E+01 ± 5.08E-01	+	1.18E+01 ± 1.34E+00	+	**1.03E+01 ± 1.12E+00**	1.37E+01 ± 1.71E-01	+	1.04E+01 ± 7.84E-01	=
f_{21}	**3.00E+02 ± 8.32E+01**	=	3.00E+02 ± 8.77E+01	=	3.00E+02 ± 8.04E+01	3.00E+02 ± 4.26E+01	=	3.00E+02 ± 6.78E+01	+
f_{22}	**1.40E+02 ± 5.62E+01**	−	1.29E+03 ± 5.41E+02	+	1.66E+02 ± 2.01E+02	4.55E+03 ± 3.33E+02	+	1.47E+03 ± 7.88E+02	+
f_{23}	6.67E+03 ± 1.24E+03	+	4.52E+03 ± 7.48E+02	=	3.97E+03 ± 1.14E+03	8.04E+03 ± 2.95E+02	+	**3.60E+03 ± 6.42E+02**	=
f_{24}	2.19E+02 ± 6.43E+00	=	2.40E+02 ± 1.15E+01	+	2.16E+02 ± 7.23E+00	3.01E+02 ± 2.62E+00	+	**2.09E+02 ± 3.28E+00**	−
f_{25}	**2.69E+02 ± 8.14E+00**	=	2.92E+02 ± 1.11E+01	+	2.80E+02 ± 3.02E+01	3.02E+02 ± 2.63E+00	+	2.73E+02 ± 6.39E+00	=
f_{26}	2.00E+02 ± 3.58E-03	=	2.00E+02 ± 7.16E-03	=	**2.00E+02 ± 8.89E-03**	2.79E+02 ± 3.73E+01	+	2.00E+02 ± 9.65E-03	=
f_{27}	4.94E+02 ± 6.56E+01	=	8.49E+02 ± 1.27E+02	+	5.28E+02 ± 1.03E+02	1.32E+03 ± 1.91E+01	+	**4.56E+02 ± 7.11E+01**	=
f_{28}	**3.00E+02 ± 4.14E-13**	−	3.00E+02 ± 1.04E-10	−	3.00E+02 ± 1.02E-07	3.00E+02 ± 1.66E-08	+	3.00E+02 ± 1.87E-09	−

The results show that MsDE-CM is significantly better than basic DE and MsDE-CB, and on average better than SaDE and MsDE-Sam.

5 Conclusions

In this work, we propose the Multi-strategy Differential Evolution (MsDE) algorithm to construct and maintain an ensemble set of strategies with various parameters. MsDE is capable of self-adapting the type of strategy and its parameters F and CR. Different from the alternative approaches, MsDE represents the ensemble strategy population as agents that interact with the candidate solutions. The performance of the strategies is measured by a performance measure which is used to self-adapt the ensemble population.

We propose two performance measures, and compare their efficiency in constructing an ensemble of successful strategies. Our results show that favoring strategies that can produce diverse trial vectors successfully yields better than favoring them based solely on their ratio of producing successful trial vectors.

We propose three approaches for self-adapting the strategy population. The simplest approach is based on random sampling where new strategies are randomly introduced into, and the ones that do not satisfy a performance criterion are removed from the ensemble. Other two approaches use clonal selection mechanism to proliferate successful strategies in the ensemble. While, four different types of strategies with their continuous F and CR parameters are aimed to be optimized, the sampling based approach requires only a parameter for ensemble size, and a threshold for defining a successful strategy based on its performance metric. The clonal selection based algorithms introduce an additional parameter for perturbing strategies. In this work, we used asynchronous selection operator. Asynchronous selection can speed up the convergence and decrease the population diversity. We would like to examine the effect of synchronous/asynchronous update in the future work.

We compare the MsDE with basic DE and the SaDE algorithm with different combinations strategy performance measure and population adaptation schemes. Overall, our results show that the MsDE provides better results on CEC2013 benchmark functions. In future works, we will try to extend the MsDE approach to other evolutionary algorithms.

Acknowledgments. We would like to thank Dr. Samaneh Khoshrou from Eindhoven University of Technology for the informative discussion. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 665347.

References

1. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning (1989)
2. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
3. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution-an updated survey. Swarm Evol. Comput. **27**, 1–30 (2016)
4. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. **33**(1–2), 61–106 (2010)
5. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
6. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput. Surv. (CSUR) **45**(3), 35 (2013)
7. Eiben, Á.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**(2), 124–141 (1999)
8. Karafotias, G., Hoogendoorn, M., Eiben, Á.E.: Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans. Evol. Comput. **19**(2), 167–187 (2015)
9. Kramer, O.: Self-adaptive Heuristics for Evolutionary Computation, vol. 147. Springer, Heidelberg (2008)
10. De Jong, K.A.: Analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis (1975)
11. Yaman, A., Hallawa, A., Coler, M., Iacca, G.: Presenting the ECO: evolutionary computation ontology. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10199, pp. 603–619. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55849-3_39
12. Hallawa, A., Yaman, A., Iacca, G., Ascheid, G.: A framework for knowledge integrated evolutionary algorithms. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10199, pp. 653–669. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55849-3_42
13. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer Science & Business Media, Heidelberg (2006)
14. Iacca, G., Caraffini, F., Neri, F.: Multi-strategy coevolving aging particle optimization. Int. J. Neural Syst. **24**(01), 1450008 (2014)
15. Iacca, G., Mallipeddi, R., Mininno, E., Neri, F., Suganthan, P.N.: Super-fit and population size reduction in compact differential evolution. In: 2011 IEEE Workshop on Memetic Computing (MC), pp. 1–8, April 2011
16. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl. Soft Comput. **11**(2), 1679–1696 (2011)
17. Mallipeddi, R., Iacca, G., Suganthan, P.N., Neri, F., Mininno, E.: Ensemble strategies in compact differential evolution. In: 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 1972–1977, June 2011
18. Iacca, G., Neri, F., Caraffini, F., Suganthan, P.N.: A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms. In: Esparcia-Alcázar, A.I., Mora, A.M. (eds.) EvoApplications 2014. LNCS, vol. 8602, pp. 615–626. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45523-4_50

19. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791. IEEE (2005)
20. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
21. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)
22. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
23. De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Trans. Evol. Comput. **6**(3), 239–251 (2002)
24. De Castro, L.N.: Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications. CRC Press (2006)
25. Liang, J., Qu, B., Suganthan, P., Hernández-Díaz, A.G.: Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212, 3–18 (2013)
26. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bull. **1**(6), 80–83 (1945)